Pre-Execution Environment (PXE) PDK v1.3

This document contains instructions for setting up a Windows NT server to respond properly to PXE Boot PROMs supporting the UNDI API for LSA2 Version 0.98i. A detailed list of revisions is contained in Appendices G and H.

A compliant Boot PROM will be able to complete the DHCP and BINL communication, and using unicast or multicast TFTP, download a series of three images provided.

Contents

1 Test Environment Setup

This section explains how to create a PXE test environment with this PDK. We suggest you use dedicated hosts for the test servers. The test servers execute Microsoft DHCP Manager, so you should set up a test network apart from your main network, to avoid conflicting with production DHCP servers.

The test environment requires a DHCP Server program and the PXE Image Server program. The DHCP Server is responsible for allocating dynamic IP addresses. The PXE Image Server, contained in this PDK, provides the BINL, ProxyDHCP, and TFTPD services.

Your can use one of two possible configurations for your test environment: the DHCP server executes on the same host as the PXE Image Server, or on a different host from PXE Image Server. The PXE PDK includes a Proxy DHCP service whose purpose is to redirect the client from the DHCP Server to the PXE Image Server host. The ProxyDHCP service should be started only if the DHCP server is executing on a host other then the PXE Image Server. The ProxyDHCP Service must be executed on the PXE Image Server .

If, during setup, you elect to NOT incorporate the ProxyDHCP service the setup program assumes the DHCP Server and the PXE Image Server are on the same host. The setup program then defines DHCP option 60 to be "PXEClient" for the DHCP server executing on the PXE Image Server host . If you select the ProxyDHCP installation, the setup program assumes the DHCP Server is separate from the PXE Image Server and installs the ProxyDHCP as a service. The DHCP option 60, for a DHCP Server that is not executing on the same host as the PXE Image Server, must not have the DHCP Option 60 defined to "PXEClient" although it may be defined to any other value.

You need at least one client PC. Although multicast TFTP can be used with a single client, to observe the full impact of master/slave multicast TFTP you need multiple clients.

1.1 Setup Instructions – DHCP and PXE Image Server on the same host.

To create a PXE Image Server:

- Install the Windows NT Server Operating System (3.51 or later) on a PC.
- Install TCP/IP and the Microsoft DHCP Server.
- Assign a static IP address to the host, and create a valid DHCP scope on the server.

You should test that the network and DHCP are working properly by booting a DHCP client on the network (such as an NT Workstation with DHCP enabled).

Now the server is ready for the PDK to be installed.

- Copy the self-extracting executable PXEPDK13.EXE to the test server.
- Extract the setup files into a temporary directory by running the executable. This extracts SETUP.EXE and additional support files. (PXEPDK13.EXE can be extracted on a separate machine. If you do this, then copy the directory SETUP to the server. Then run SETUP.EXE, from the SETUP directory, and choose a directory to install the PDK software.)
- In This configuration you do NOT want to install the ProxyDHCP service. See the next two sections for a description of the ProxyDHCP service. The setup program installs services and sets up registry keys.
- Choose to reboot the system so the services start correctly.

The server is now ready to respond to PXE compliant client Boot PROMs.

1.2 Setup Instructions – DHCP separate from the PXE Image Server

- Create a DHCP Server by installing the Windows NT Server operating system (3.51 or later) , TCP/IP and the Microsoft DHCP Server on a host.

- Assign a static IP address to the host, and create a valid DHCP scope on the server. You should test that the network and DHCP are working properly by booting a DHCP client on the network (such as an NT Workstation with DHCP enabled).
- Ensure that the DHCP option 60 is not defined to "PXEClient".

To install the PDK on the separate PXE Image Server:

- Install Windows NT Server operating system (3.51 or later) and TCP/IP on a host.
- Copy the self-extracting executable PXEPDK13.EXE to the PXE Image Server host.
- Extract the setup files into a temporary directory by running the executable. This extracts SETUP.EXE and additional support files.
- Run SETUP.EXE, and choose a directory to install the PDK software. The setup program installs services and sets up registry keys.
- Choose to reboot the system so the services start correctly.

The server is now ready to respond to PXE compliant client Boot PROMs.

1.3 PXE Client Setup

Next, you should set up a client PC that supports either Plug-and-Play/BIOS Boot Specification or network boot devices that hook interrupt 18h. To test the working Boot PROM images included with this PDK, you need an Intel EtherExpress Pro/100B (E100B) network interface card (NIC), or an E100B on the motherboard. You can write the image to the E100B's FLASH memory using the included utility FUTIL.EXE. The utility is documented in the table below. Run the program on the client machine, passing the name of the appropriate binary image file.

You can program the PXE Boot PROM code, e100b.nic, into the FLASH memory on an E100B card using the command:

```
futil e100b.nic
```

You can incorporate the BIOS-based PXE Boot PROM code (E100B.LD) into your client two ways. The first way is to program the image into the FLASH memory on an E100B card using the command:

```
futil e100b.ld
```

The second way is to include the BIOS-based image (E100B.LD) as an option ROM image in your system BIOS image (as you would include a video option ROM image). You must use your own BIOS programming utility to program this BIOS image into the motherboard's FLASH memory.

You may need to modify the system's CMOS settings so that the system boots from the network before going to the local disk.

FUTIL.EXE documentation.
(This is displayed by typing: FUTIL /?)

```
[FUTIL ver 2.11] - Intel PCI NIC FLASH Update Utility
Copyright (C) 1995,1996  Intel Corporation.  All rights reserved.

Usage: FUTIL options command/filename

Options:

     -A20=#
          This option forces the use of a specific A20 gate service.
          If this option is not specified, the A20 gate service will
          be determined through software.
               A20 services: 1=XMS, 2=Int15, 3=Port92

     -bus=# -dev=# -func=#
          Specify PCI bus/device/function numbers of Intel PCI
          network adapter to be programmed.  If not specified,
          -bus, -dev and -func will default to all detected
          Intel PCI network adapters.

Commands:

     -erase
          Erases the contents of the FLASH memory.

     Filename
          Program a raw binary image into the FLASH memory.

Exit codes:
     0 := All FLASH operations completed successfully.
     1 := Bad command line parameter.
     2 := No supported Intel PCI network adapters detected.
     3 := No supported FLASH devices detected.
     4 := FLASH operation failed
     5 := Image file is missing or corrupted.
```

2 PXE Image Server Overview

For detailed information on the requirements for creating a PXE Boot PROM, see the Network PC (Net PC) design guidelines[1] included with this PDK. The following is an overview of the files included in this PDK and their relationship to the Net PC design guidelines.

2.1 BINL Service Overview

This PDK includes a BINL (Boot Image Negotiation Layer) service and a related DLL. A PXE compliant Boot PROM sends a BINL Request packet after accepting an IP address from a DHCP server. It discovers the IP address of the BINL server through the DHCP redirection process. BINL currently listens on UDP port 4011 as assigned by IANA.

2.2 TFTPD Service Overview

This PDK includes a TFTPD service that provides unicast and multicast TFTP transfer of image files. After receiving a boot file name from BINL, a PXE Boot PROM downloads the file from TFTPD and executes it. The unicast TFTP service is provided on the standard well-known port 69. Multicast service is provided on a configurable port; registry settings created by this PDK set the port to 1759.

The TFTPD service provides two options for determining packet sizes; one for multi-cast and one for uni-cast. Under multi-cast the server will send files using a packet size of 1456 bytes if the MTFTPD value *MTFTP_HIGH_PERFORMANCE* is set to 1. If *MTFTP_HIGH_PERFORMANCE* is set to 0 512 byte packets are used. Note that this size applies to all clients. Therefore, all of your clients must be capable of handling the larger packet size to use in increased performance. Otherwise, set *MTFTP_HIGH_PERFORMANCE* zero.

For unicast TFTP, the client can negotiate a packet size from 512 bytes to 2560 bytes. The client must use the TFTP block size option as specified in RFC 1783.

2.3 PXE Image Files Overview

This PDK includes sample image files named BSTRAP.1, TEST.2, TEST.3, DOSUNDI.2 and DOSUNDI.3. The BSTRAP.1 image is the bootstrap file specified in a BINL Reply and downloaded and executed by the client. This first image presents a menu of the available management and OS boot options, according to the information received in the BINL Reply. In the default scenario installed by the PDK, the menu consists of choices for executing the API test and for logging into the server. Pressing Esc or allowing the menu to timeout causes the system to boot from the next boot device specified in the system CMOS..

When the test boot is chosen, the second image (TEST.2) is downloaded and executed. This prepares an environment, in which a DOS image can run, and downloads the much larger TEST.3 DOS test image. The included version of the TEST.3 image contains the PXETEST.EXE program, which tests the Boot PROM's API implementation, and is run automatically by the image's AUTOEXEC.BAT file. More details on the PXETEST.EXE program can be found in the downloaded TEST.3 image in PXETEST.TXT.

When the login boot is chosen, the second image (DOSUNDI.2) is downloaded and executed. This prepares an environment, in which a DOS image can run, and downloads the much larger DOSUNDI.3 DOS image. The included version of the DOSUNDI.3 image executes net logon in the AUTOEXEC.BAT and attempts to log the client into the server. If the logon is successful you will be able to map server drives to the client. To thoroughly test your system you should attempt to execute sizable applications and observe characteristics like memory usage.

[1] *Network PC System Design Guidelines: A Reference for Designing Net PC Systems for Use with t.he Microsoft Windows and Windows NT Operating Systems*. Included in the PDK as NETPC.DOC.

2.4 DHCP Server Overview

If the DHCP Server is on the same host as the PXE Image Server, a Boot PROM cannot determine if it has received PXE management information in a DHCP_OFFER packet unless that packet reflects the DHCP Option 60 (Class Identifier) containing "PXEClient" which the Boot PROM sent in the DHCP_DISCOVER packet. If the DHCP Server does not send this option, the Boot PROM must exhaust the complete Proxy Timeout waiting for such a packet. Once the timeout is finished, the Boot PROM assumes there is a BINL server on the DHCP server machine, and tries to send a BINL request to it. In the Net PC design guidelines, we recommend the Proxy Timeout employ an exponential back-off timeout totaling 28 seconds. To avoid this delay in connecting to the BINL server when there is no Proxy DHCP Server, the DHCP Server must reflect the Option 60 "PXEClient".

The PDK installation adds option 60 to the DHCP Server, by modifying registry values. It contains nine bytes of data that read “PXEClient” in ASCII (no null terminator). Once these registry keys are modified, the DHCP Server sends the Class Identifier option "PXEClient" to all clients which request option 60. That is, for a Boot PROM to receive this option from the NT DHCP Server, it has to specifically request it using the PREQLST (parameter request list) DHCP option (see RFC 1533). This is not part of the Net PC specification for a PXE Boot PROM but an optimization for Boot PROMs that will be used with the NT DHCP Server.

If the DHCP Server is on a different host than the PXE Image Services, the Proxy DHCP responds to the client with the proper IP address for the PXE Image Server. After accepting an IP address from the DHCP service the client should then send a BINL request to the PXE Image Server using the IP address provided by the Proxy DHCP response.

3 Understanding the Boot PROM Communication

The Intel LCM BINL service, referred to as BINL, provides the NIC client with a menu of images that can be downloaded and the information needed to make a connection with the multicast TFTP server. BINL also provides either the server's workgroup or domain name that can be retrieved by the GET_BINL_INFO API. The information provided by BINL is read from registry keys. There is a detailed illustration of these registry keys in Appendices A and B of this document. Also, the file pkttrace.txt included with this PDK contains a complete trace of the network packets as captured by a packet snifter during a client boot.

3.1 BINL Reply Packet

The following table (see Table 3.1) shows the layout of the packet that would be returned by the BINL included in this version of the PDK. This table is only provided for illustration. The actual options in the table are subject to change and may occur in a different sequence then what is illustrated.

BINL sends the following categories of vendor specific information:

- sub-directory path for the NIC specific image files (PXE_NIC_PATH);
- management menu data (PXE_MAN_MENU);
- OS menu options (PXE_OS_MENU);
- MTFTP data (MTFTP IP Addr, MTFTP Client UDP, MTFTP Server UDP, MTFTP Start Delay, MTFTP Timeout Delay); and
- Server identification information (PXE_LCMSERVER, PXE_LCMWRKGRP); and
- GUID, when none is sent by the client.

The management menu option (PXE_MAN_MENU) contains the data needed to download the image that is to perform the management function. In the PDK this image is a test program.

The OS menu options (PXE_OS_MENU) would be additional menu options that could be displayed for user selection.

The MTFTP options provide the client the information needed to download the initial boot strap program whose filename is contained in the "bootfile" field of the fixed portion of the BINL DHCP packet.

The server identification information (PXE_LCMSERVER, PXE_LCMDOMAIN, and PXE_LCMWRKGRP) is used to identify the work group or domain of the responding server for processes that may need that information to establish communication with the server.

3.2 Forming Paths to Image Files

Three pieces of data are required to form a full path to the NIC specific image files: the bootfile directory from the "bootfile" field, the NIC sub-directory and the filename. The "bootfile" field contains the full path and filename of the bootfile. To create the base directory, strip the filename from the bootfile path. Then append to the base directory the NIC specific directory from the PXE_NIC_PATH option (see Table 3.1). Finally the image filename is formed and appended to the NIC specific directory just formed. The image filename is formed by extracting the base file name from either the PXE_MAN_MENU option or one of the PXE_OS_MENU options. The base file name then has the file number added as an extension. For example, TEST becomes TEST.2 or TEST.3.

For instance, for a UNDI compliant boot PROM on an X86 computer the "bootfile" field might contain the value:

D:\Program Files\Intel\PXE_PDK\System\Images\X86PC\bstrap.1

The PXE_NIC_PATH option would have the value: UNDI

The PXE_OS_MENU option might have the value:

```
2,224.0.0.2,224.0.0.3,TEST,UNDI API TEST
```

The final path to the TEST.2 file would be:

```
D:\Program Files\Intel\PXE_PDK\System\Images\X86PC\UNDI\TEST.2
```

Table 3.1: BINL DHCP OPTIONS

DHCP Options				
Tag Name	***Tag Number***	***Length Field***	***Data***	**Description**
			99.130.83.99	Magic Cookie
DHCP Message Type	53	1	4	DHCP ACKNOWLEDGE
Client Class	60	9	"PXEClient"	Indicates this packet is from an LSA2 client.
DHCP UUID/GUID	97	17	Varies	A UUID/GUID for the client.
DHCP VENDOR	43	6		Start of PXEClient NIC Path
PXE_NIC_PATH	64	5	"UNDI\0"	Sub-directory for images for UNDI clients from registry
PXE_END	255			End of Option
DHCP VENDOR	43	36		Start of Management Menu
PXE_MAN_MENU	65	35	2 224.0.0.2 224.0.0.3 TEST\0 PXE BootPROM Test Kit\0	The number of files with this base name that are to be downloaded. Also the number of IP addresses contained in this sub-option. The base name of the files. The text to be displayed in the menu. These items are concatenated into this field with the number of files in binary, the IP address in net format, and the strings null terminated.
PXE_END	255			End of Option
DHCP VENDOR	43	1		OS Menu Options This option would normally contain the entries for OS menu choices from the registry. These entries would be similar to the menu entry above. Because this test version is shipped without any OS menu choices defined this option is empty but is here as a placeholder.
PXE_OS_MENU	66	35	2 224.0.0.2 224.0.0.3 TEST\0 PXE BootPROM Test Kit\0	The number of files with this base name that are to be downloaded. Also the number of IP addresses contained in this sub-option. The base name of the files. The text to be displayed in the menu. These items are concatenated into this field with the number of files in binary, the IP address in net format, and the strings null terminated.
PXE_OS_MENU	66	35	2 224.0.0.4 224.0.0.5 DOSUNDI\0 Remote Logon to the Server\0	The number of files with this base name that are to be downloaded. Also the number of IP addresses contained in this sub-option. The base name of the files. The text to be displayed in the menu. These items are concatenated into this field with the number of files in binary, the IP address in net format, and the strings null terminated.
PXE_END	255			End of Option
DHCP VENDOR	43	21		MTFTP Options (From registry)
MTFP IP Addr	1	4	224.0.0.2	MTFTP IP address for BSTRAP.1
MTFTP Client UDP	2	2	1758	Client MTFTP Port
MTFTP Server UDP	3	2	1759	Server MTFTP Port
MTFTP Start Delay	4	1	1	
MTFTP Timeout Delay	5	1	2	
PXE_END	255			End of Option
SERVER IDENTIFIER	54	4	Varies	IP address of responding server
DHCP VENDOR	43	Varies		Server Identifier Options
PXE_LCMSERVER		Varies	??	Name of the server executing BINL
PXE_LCMWRKGRP		14	LCM_WORKGROUP	Normally would be the name either the domain or the workgroup for the server from the registry. This test release sets it to "LCM_WORKGROUP".
PXE_END	255			End of Option

4 Modifying the Image Structure

The BINL service is capable of displaying a menu of management and OS boot options. This section deals with how you can create new images and have BINL provide them as options for clients.

4.1 Creating New DOS Images

The utility MKIMAGE.EXE is included in the PDK, to create a DOS image (such as TEST.3). To use this utility, you need to create a 1.44 MB DOS boot diskette (using the DOS sys command or format /s). You can leave the diskette as it is (containing only the MSDOS.SYS, IO.SYS, and COMMAND.COM files), or add AUTOEXEC.BAT and executables, etc. Whatever you put on the disk becomes part of the image. For instance, you can include the PXETEST.EXE like the sample TEST.3 image included in the PDK.

Place this diskette in the A: drive of a machine and run MKIMAGE.EXE from another drive. This writes an image of the A: diskette into the current directory in a file named TEST.BIN. You should rename this file to TEST.3 and place it in the appropriate directory (see Appendix B, **Directory Structure**, below).

4.2 Adding Menu Options

Adding menu options to provide the user with additional images to download is a two part process. First, the menu options are added to the MENU value in the X86PC\UNDI key. Second, the MTFTPD\FILES registry key must be updated with the new filenames. The TFTPD service automatically assigns IP addresses to the files when the system reboots.

The MENU value can be edited using the registry editor. Before making any changes to the Registry you should save the current copy so you can recover from any errors. The class of the MENU is REG_MULTI_SZ. Each string in MENU makes up a complete menu option entry. Each menu option in MENU consists of the number of files to be downloaded, the base file name, and the text to be displayed on the menu option. Each image to be downloaded can have multiple parts and the filenames are formed by appending ".2", ".3", etc. to the base filename indicated by the menu option. For example the entry for an image to load a xYz OS which has image files xYz.2 and xYz.3 might look like this:

2,xYz,Load xYz OS

Note that the fields in the string are comma delimited.

The second part of adding menu options is to ensure the files have corresponding value entries in the MTFTPD\FILES registry key. These values are type REG_SZ and hold the IP address that MTFTPD uses to multicast transmit the file to the clients. The entries in these values consist of a full path and filename for the value name and an IP address for the value data. The IP address is generated by MTFTPD automatically when the system reboots. Typical examples of these values, as viewed with the registry editor, would look like this:

C:\Intel\BootPROM\PXE_PDK\X86PC\UNDI\xYz.2:REG_SZ:224.0.0.9

C:\Intel\BootPROM\PXE_PDK\X86PC\UNDI\xYz.3:REG_SZ:224.0.0.10

Once new values are added to the MTFTPD\FILES registry key, you must stop and restart the Intel LCM TFTPD service either by using the Services applet or rebooting the system.

4.3 Rules for Displaying Menu Options

To minimize unnecessary user input the following rules govern display of the menu options for the boot process.

- If there is only a management option as indicated by the MAN value in BINL key, then the user has three options: press space bar to execute the management option, press ESC to perform a local boot, or let the menu timeout and perform a local boot.

- If there is a management option and only one MENU option the user has these three choices: press space bar to execute the management option, press ESC to perform a local boot, or let the menu timeout and execute the MENU option. Note that with only one MENU option the user does not see a menu since the menu would have only one option to choose.
- If there is a management option and multiple menu options then the user has these four choices: press space bar to execute the management option, press ESC to perform a local boot, let the menu timeout and execute the default menu option, or chose a menu option to execute.

Appendix A: Registry Entries

This appendix explains most of the registry keys that the PDK creates on the test server. Under HKEY_LOCAL_MACHINE\SOFTWARE, there is a key named "Intel", and under that a key named "LANDesk". The following subkey hierarchy is created under LANDesk.

A.1 LCM_SYSTEM Key

SOFTWARE\Intel\LANDesk\LCM_SYSTEM

The values within the LCM_SYSTEM key specify information about the NT network environment of the test server on which this software is being installed.

Values:

DomainName : REG_SZ	Name of Domain or Workgroup to which the test server belongs.	(e.g. LCM_WORKGROUP)
IsDomain : REG_DWORD	1 if the test server belongs to a Domain, 0 if it belongs to a Workgroup.	(e.g. 0)

A.2 BINL Key

SOFTWARE\Intel\LANDesk\LCM_SYSTEM\BINL

The values within the BINL Key specify information about the TFTP server, which is independent of architecture or NIC type.

Values:

IMAGE_PATH : REG_SZ	Base directory to the image files. (Case sensitive; must correspond to MTFTPDIR.TXT file entries).	(e.g. C:\LCM\SYSTEM\IMAGES)
PROC_ARCH: REG_MULTI_SZ	Array of processor architectures indexed by architecture number from DHCP packet The text in this field corresponds to the keys below and to the directory structure.	X86PC

Subkeys:

The BINL key contains architecture specific keys that specify run time values unique to each processor architecture, such as \X86PC.

A.3 Architecture Specific Keys

SOFTWARE\Intel\LANDesk\LCM_SYSTEM\BINL\X86PC

Within each of the Architecture Keys is a value for the initial boot filename and the MTFTP IP address for the boot file. Each Architecture Key also contains NIC Keys for each of the NIC types to be supported within the architecture.

Values:

BOOTFILE: REG_SZ	Contains the path and filename of the initial boot file, and the file's associated MTFTP IP address. The path is relative to the base image directory (specified by IMAGE_PATH above). (Case sensitive; must correspond to MTFTPDIR.TXT file entry).	X86PC\BSTRAP.1

Subkeys:

\UNDI

\PCI-8086-1229-020000-01

\PNP-8130-020000

A.4 NIC Specific Keys

SOFTWARE\Intel\LANDesk\LCM_SYSTEM\BINL\X86PC\UNDI

The values within the NIC-Specific Keys define the base filenames and text for the management and OS menu options.

Values:

MAN:REG_SZ:	Information for Boot PROM test option. .	2,TEST,PXE Boot PROM PDK
MENU: REG_MULTI_SZ:	Information for OS menu choices. This value must exist, but can be empty if no OS images are available.	2,NT50,Windows NT Version 5.0 2,xYz,xYz OS 1.0

A.5 MTFTPD Key

SOFTWARE\Intel\LANDesk\LCM_SYSTEM\MTFTPD

The values within the MTFTPD Key specify information about the TFTP server, which is independent of architecture or NIC type.

Values:

MCAST_ENABLE	Indicates whether multi-cast TFTP is enabled.	1
MCAST_HIGH_PERFORMANCE : REG_DWORD	Indicates whether MTFTPD should send files using the larger 1456 byte packets or the smaller 512 byte packet.	1 – (Default) Send files with 1456 byte packets 0 – Send files with 512 byte packets
MCAST_CLNT_PORT : REG_SZ	Client port to MTFTP	1758
MCAST_OPN_TO: REG_SZ	Open time out for client	1
MCAST_REOPN_TO: REG_SZ	Time out before reopening MTFTP file transmission.	2
MCAST_SRVR_PORT: REG_SZ	Server port for MTFTP (specified on command line when starting MTFTP server.)	1759
MCAST_START_ADDRESS	Beginning of the range of IP addresses to be assigned to the multi-cast files.	224.0.0.2

Subkeys:

The MTFTPD key contains one sub-key name FILES that maps the multicast filenames to the corresponding IP addresses.

A.6 FILES Keys

SOFTWARE\Intel\LANDesk\LCM_SYSTEM\MTFTPD\FILES

Within the FILES key is a set of values that map filenames to IP addresses for multicast FTP. The name of the value is the full pathname of the file while the data for the value is the multicast IP address for the file. The data for the values is determined dynamically by the TFTP service whenever the service is started or stopped. You may create an entry manually but the next time the TFTP service is started it may overwrite the IP address you specify with a new IP address.

Values: (These vary depending on the installation directory.)

D:\Program Files\Intel\PXE_PDK\System\Images\X86PC\BSTRAP.1: REG_SZ	Contains the file's associated MTFTP IP address.	varies
D:\Program Files\Intel\PXE_PDK\System\Images\X86PC\TEST.2: REG_SZ	Contains the file's associated MTFTP IP address.	varies
D:\Program Files\Intel\PXE_PDK\System\Images\X86PC\TEST.3: REG_SZ	Contains the file's associated MTFTP IP address.	varies

Subkeys:

None

Appendix B: Sample Registry Structure

Items marked with an asterisk are representative of actual values that are to be determined when the PDK is installed.

```
Key Name:          SOFTWARE\Intel
Class Name:        GenericClass
Last Write Time:   6/10/97 - 10:03 PM
Key Name:          SOFTWARE\Intel\LANDesk
Class Name:        <NO CLASS>
Last Write Time:   6/4/97 - 7:45 AM
Key Name:          SOFTWARE\Intel\LANDesk\LCM_SYSTEM
Class Name:        <NO CLASS>
Last Write Time:   6/10/97 - 1:01 PM
Value 0
  Name:            DomainName
  Type:            REG_SZ
  Data:            LCM_WORKGROUP*
Value 1
  Name:            IsDomain
  Type:            REG_DWORD
  Data:            0*
Value 2  Name:            ServerName
  Type:            REG_SZ
  Data:            PXE_PDK_SERVER*
Key Name:          SOFTWARE\Intel\LANDesk\LCM_SYSTEM\BINL
Class Name:        <NO CLASS>
Last Write Time:   6/4/97 - 7:51 AM
Value 0
  Name:            IMAGE_PATH
  Type:            REG_SZ
  Data:            D:\Program Files\Intel\PXE_PDK\System\Images*
Value 1
  Name:            PROC_ARCH
  Type:            REG_MULTI_SZ
  Data:            X86PC
Key Name:          SOFTWARE\Intel\LANDesk\LCM_SYSTEM\BINL\X86PC
Class Name:        <NO CLASS>
Last Write Time:   6/4/97 - 7:51 AM
Value 0
  Name:            BOOTFILE
  Type:            REG_SZ
  Data:            X86PC\BSTRAP.1
Key Name:          SOFTWARE\Intel\LANDesk\LCM_SYSTEM\BINL\X86PC\UNDI
Class Name:        <NO CLASS>
Last Write Time:   6/4/97 - 7:51 AM
Value 0
  Name:            MAN
  Type:            REG_SZ
  Data:            2,Test,PXE PDK Test Programs
Value 1
  Name:            MENU
  Type:            REG_MULTI_SZ
  Data:
Key Name:          SOFTWARE\Intel\LANDesk\LCM_SYSTEM\MTFTPD
Class Name:        <NO CLASS>
Last Write Time:   6/10/97 - 10:47 AM
Value 0
  Name:            MCAST_CLNT_PORT
  Type:            REG_SZ
  Data:            1758
Value 1
  Name:            MCAST_ENABLE
  Type:            REG_DWORD
```

```
  Data:            0x1
Value 2
  Name:            MCAST_OPN_TO
  Type:            REG_SZ
  Data:            1
Value 3
  Name:            MCAST_REOPN_TO
  Type:            REG_SZ
  Data:            2
Value 4
  Name:            MCAST_SRVR_PORT
  Type:            REG_SZ
  Data:            1759
Value 5
  Name:            MCAST_START_ADDRESS
  Type:            REG_SZ
  Data:            224.0.0.2
Value 6
  Name:            MCAST_HIGH_PERFORMANCE
  Type:            REG_DWORD
  Data:            1
Key Name:          SOFTWARE\Intel\LANDesk\LCM_SYSTEM\MTFTPD\FILES
Class Name:        <NO CLASS>
Last Write Time:   6/10/97 - 10:47 AM
Value 0
  Name:            D:\Program Files\Intel\PXE_PDK\System\Images\X86PC\BSTRAP.1*
  Type:            REG_SZ
  Data:
Value 1
  Name:            D:\Program Files\Intel\PXE_PDK\System\Images\X86PC\UNDI\TEST.2*
  Type:            REG_SZ
  Data:
Value 2
  Name:            D:\Program Files\Intel\PXE_PDK\System\Images\X86PC\UNDI\TEST.3*
  Type:            REG_SZ
  Data:
Key Name:          SOFTWARE\Intel\LANDesk\LCM_SYSTEM\PROXYDHCP
Class Name:        <NO CLASS>
Last Write Time:   6/4/97 - 7:49 AM
Value 0
  Name:            IMAGE_PATH
  Type:            REG_SZ
  Data:            D:\Program Files\Intel\PXE_PDK\System\Images*
Value 1
  Name:            MAGIC_PROXYDHCP_COOKIE
  Type:            REG_SZ
  Data:
```

Appendix C: Image Directory Structure

Images Base Directory — defined by the registry key/value:
HKEY_LOCAL_MACHINE\SOFTWARE\Intel\LANDesk\LCM_SYSTEM\BINL\IMAGE_PATH:REGSZ:

base_image_directory

Typical Value: C:\Program Files\Intel\PXE_PDK\System\Images

The Images Base Directory contains an Architecture Subdirectory for each supported processor architecture, such as \X86PC.

Each Architecture Subdirectory contains a NIC Directory for each NIC type with an OS supported for that processor architecture and the actual bootstrap file for that processor architecture.

Typical Values:

BSTRAP.1

\UNDI

\PCI-8086-1229-020000-01

\PNP-8130-020000

Each NIC Directory contains the management files as well as all of the OS files for that NIC/Processor combination.

Table C.1: **Sample Directory Structure**

C:\LCM\SYSTEM\IMAGES
\X86PC
BSTRAP.1
\UNDI
TEST.2
TEST.3

Appendix D: SYSID BIOS Support Interface Requirements

D.1 Introduction

The following information is intended to provide the details necessary to provide the required System BIOS interfaces for Intel's Desktop Management BIOS (hereafter DMI BIOS) implementation. Thus, this document can be used as a guide, along with the other referenced DMI documents, to aid DMI Instrumentation efforts.

D.2 Definition of Terms

The following information defines the terms used in this document.

DMI	Desktop Management Interface— See "Desktop Management BIOS Specification" Version 2.0 dated October 16, 1995 for more details.
IETF	Internet Engineering Task Force. Also see the following definition of Internet Draft.
INTERNET-DRAFT	Internet-Drafts are documents of the IETF and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts may be updated, replaced, or obsolete at any time. It is inappropriate to use Internet Drafts as reference material or to cite them other than as "work in progress." To learn the current status of any Internet Draft, please check the "1id-abstracts.txt" listing contained in the Internet Draft Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).
NVRAM	Non-Volatile RAM—A type of storage device used to retain programmed information after power is removed from the CPU and Main Memory.
SYSID	A superset of identifiers that allows definition and organization of unique identifiers. This superset is comprised of identifiers such as a UUID and 1394 GUIDs. A UUID is a SYSID, but a SYSID need not be a UUID (see the following definition of a UUID).
SYSTEM EVENT LOG	A log file utilized to store critical system status in an area of Non-Volatile RAM.
UTC	Coordinated Universal Time.
UUID and GUID	Universally Unique Identifier and Globally Unique Identifiers—A UUID or GUID is 128 bits long, and if generated according to the one of the mechanisms in this document, it is designed to be different from all other UUIDs/GUIDs generated until 3400 A.D., or extremely likely to be different (depending on the mechanism chosen).

D.3 Required Interfaces

This document is required to completely define the interfaces. Thus, the following list attempts to explain in short concise statements the interface requirements of the SYSID BIOS support. Any additional clarification follows this section.

1. Support in System BIOS to provide full DMI compliance according to the Version 2.0 specification referred to above.
2. Support in System BIOS to provide SYSID capability. One such SYSID provision is the UUID/GUID capability at a motherboard granularity level. This support is based upon the IETF Internet-Draft titled "UUIDs and GUIDs" by Paul J. Leach of Microsoft and Rich Salz of the Open Group. The intent is to provide a mechanism for the storage of a UUID without the BIOS being required to actually generate the UUID.

D.4 SYSID BIOS Table Interface

This SYSID BIOS interface takes into account the support of Operating Systems that do not provide easy software access to the existing programmatic BIOS interfaces. One example is Windows NT 3.51 or later.

There are two distinct interfaces required to provide this support:

1. Support in System BIOS to provide a Table of SYSID BIOS Structure information that is compiled at POST Time and static at Run-Time. This is referred to as the "SYSID BIOS Structure Table Interface" and is described below.
2. Additional details on SYSID BIOS Initialization/Reprogramming support that is different for Operating Systems that fit this model.

D.4.1 SYSID BIOS Structure Table Interface

The interface required to provide SYSID BIOS structure table interface support is described here:

1. **Entry Point Structure—**This is found in the 000E0000h to 000FFFFFh physical address area of Memory/RAM. ***The Entry Point Structure is PARAGRAPH Aligned!!***

Entry Point Structure

ELEMENT	LENGTH	DESCRIPTION
Header/Type	7 Bytes	_SYSID_
Checksum	1 Byte	Checksum of the SYSID BIOS Entry Point Structure
Length	2 Bytes	Total length of SYSID BIOS Structure Table (Set to 011h).
SYSID BIOS Structure Table Address	4 Bytes	32 bit physical address of the beginning of the SYSID BIOS Structure Table. ***This value is BYTE Aligned!!***
Number of SYSID BIOS Structures	2 Bytes	Total number of structures within the SYSID BIOS Structure Table.
SYSID BIOS Revision	1 Byte	Revision of the SYSID BIOS Extensions (Set to 00h).

Header/Type—This is a fixed size for this entry point structure. It is always seven bytes long. The first and last byte are always the underscore ASCII characters. The middle five bytes are the ASCII characters of SYSID.

Checksum—This value is a two's complement based checksum which causes the addition of all bytes defined for this table interface to be equal to 00h. Please note that this is a 8-bit addition calculation (byte wide addition).

Length—This value is a Total length of this entire SYSID BIOS entry point structure. In other words, this value is the addition of all the bytes in this structure from the first byte of the Header/Type field to the last byte in the SYSID BIOS Revision field. This value is always 011h

Number of SYSID BIOS Structures—This is intended to allow easy expansion of this interface should more than one type of SYSID be required. This is a integer value in hex representation. Thus, if a single SYSID structure is included, this value is 0001h.

SYSID BIOS Revision—This value is used to define modification to the SYSID BIOS Structure Table Interface. A value of 00h refers to the initial revision of this header for backward compatibility. A value of 01h refers to the next version header, etc. Current revision is 00h.

Revision 00h translates into two defined and supported SYSID BIOS Structures. These structures are provided to support:

- the IETF Internet-Draft on "UUIDs and GUIDs" by Paul J. Leach of Microsoft and Rich Salz of the Open Group
- 1394 IDs.

2. **SYSID BIOS Structure Table—**This area is pointed to by the "SYSID BIOS Structure Table Address" element (which is part of the "Entry Point Structure"). The SYSID BIOS Structure Table contains all of the SYSID BIOS structures fully packed together. The SYSID BIOS structures in this table can then be parsed. Note that every supported SYSID BIOS structure uses the template in the following table**:**

SYSID Structure Format

ELEMENT	LENGTH	DESCRIPTION
Header/Type	6 Bytes	_????_
Checksum	1 Byte	Checksum of the SYSID BIOS Structure
Length	2 Bytes	Total length of SYSID BIOS Structure
Variable Data Portion	?? Bytes	Depends on SYSID BIOS Structure Header/Type Field

Header/Type—This is a fixed size for all SYSID BIOS Structure Types. It is always 6 bytes long. The first and last byte are always the underscore ASCII characters. The middle four bytes are variable depending on the type of SYSID BIOS Structure. The currently supported types are defined below.
Checksum—This value is a two's complement based checksum which causes the addition of all bytes defined for this table interface to be equal to 00h. Please note that this is a 8-bit addition calculation (byte wide addition).
Length—This value is a Total length of the entire SYSID BIOS Structure type. In other words, this value is the addition of all the bytes in this structure from the first byte of the Header/Type field to the last byte in the Variable Data Portion field. The minimum value for this field (for one byte in the Variable Data Portion) is 0Ah. The maximum value is 0FFFFh.
Variable Data Portion—This value is entirely dependent on the Header/Type of SYSID BIOS Structure. The minimum size of this field is 01h bytes. The maximum size of this field is 0FFF6h. The size of 00h is invalid.

D.5 Types of SYSID BIOS Structures Supported

This section details the types of SYSID BIOS Structures that are currently supported in Revision 00h.
Revision 00h translates into two defined and supported SYSID BIOS Structures. These structures are provided to support:

- the IETF Internet-Draft on "UUIDs and GUIDs" by Paul J. Leach of Microsoft and Rich Salz of the Open Group
- 1394 IDs.

D.5.1 UUID Type

UUID Structure Format

ELEMENT	LENGTH	DESCRIPTION
Header/Type	6 Bytes	_UUID_
Checksum	1 Byte	Checksum of the UUID BIOS Structure
Length	2 Bytes	Total length of UUID BIOS Structure (Set to 0019h).
Variable Data Portion	16 Bytes	Actual UUID data (Initially set all bytes to 0FFh).

Header/Type—This is a fixed size for all SYSID BIOS Structure Types. It is always 6 bytes long. The first and last byte are always the underscore ASCII characters. The middle four bytes are the ASCII characters of UUID.
Checksum—This value is a two's complement based checksum which causes the addition of all bytes defined for this table interface to be equal to 00h. Please note that this is a 8-bit addition calculation (byte wide addition).
Length—This value is a Total length of the entire UUID BIOS Structure type. In other words, this value is the addition of all the bytes in this structure from the first byte of the Header/Type field to the last byte in the Variable Data Portion field. The value for this field (for 16 bytes in the Variable Data Portion) is 019h.
Variable Data Portion—This value is entirely dependent on the Header/Type of SYSID BIOS Structure. This field is a length of 10h bytes and the values are defined and programmed by utilizing the "SYSID Programming Interface Specification." The System BIOS does not program these values. The initial value provided consists of 10h bytes of 0FFh.

D.5.2 1394 ID Type

1394 ID Structure Format

ELEMENT	LENGTH	DESCRIPTION
Header/Type	6 Bytes	_1394_
Checksum	1 Byte	Checksum of the 1394 BIOS Structure.
Length	2 Bytes	Total length of 1394 BIOS Structure (Set to 0011h).
Variable Data Portion	8 Bytes	Actual 1394 ID data (Initially set all bytes to 0FFh).

Header/Type—This is a fixed size for all SYSID BIOS Structure Types. It is always 6 bytes long. The first and last byte are always the underscore ASCII characters. The middle four bytes are the ASCII characters of 1394.
Checksum—This value is a two's complement based checksum which causes the addition of all bytes defined for this table interface to be equal to 00h. Please note that this is a 8-bit addition calculation (byte wide addition).
Length—This value is a Total length of the entire 1394 ID BIOS Structure type. In other words, this value is the addition of all the bytes in this structure from the first byte of the Header/Type field to the last byte in the Variable Data Portion field. The value for this field (for 8 bytes in the Variable Data Portion) is 011h.
Variable Data Portion—This value is entirely dependent on the Header/Type of SYSID BIOS Structure. This field is a length of 08h bytes and the values are defined and programmed by utilizing the "SYSID Programming Interface Specification". The System BIOS does not program these values. The initial value provided consists of 08h bytes of 0FFh.

D.5.3 Additional Details on SYSID BIOS Support

This section describes additional points of data required when dealing with this type of Operating System Interface.

- Programming SYSID values into the BIOS is performed via the DMI BIOS Specification interface Version 2.0.
- For details in programming the SYSID values, refer to the SYSID Programming Interface Specification Revision 1.1 document contained in Appendix E.

Appendix E: SYSID Programming Interface Specification

E.1 Purpose:

This specification was developed to define a method for permanently writing and storing Universally Unique ID (UUID) or any other SYSIDs. This specification utilizes the DMI interfaces defined in the DMI BIOS specification. For definition and details on SYSIDs and UUIDs see "DMI Support Specification Version 2.2 or greater.

E.2 Reference:

Desktop Management BIOS Specification
Version 2.0
October 16, 1995

E.3 Specification:

The method of writing and storing these SYSIDs is accomplished by making a DMI BIOS call utilizing the DMI Control (54h) Function.

The format of the DMI control Function is as follows:

Function	54h
Sub Function	4007h
Data	See *Data* Structure below. Must have read \ write access.
Control	Bit 0 set to 1 (perform operation immediately).
dmiSelector	Provided from Function 50h call.
BiosSelector	Provided from Function 50h call.

Format of the *Data* parameter passed:

Offset	Name	Length	Value	Description
00h	*type*	BYTE	Varies	Type of data to be written. See type definition table below.
01h	*length*	BYTE	Varies	Length in bytes of the data to be written.
02h	*idData*	DWORD	Varies	FAR * to actual data to be written.
06h	*LpDmiWorkBuffer*	DWORD	Varies	FAR * to a readable / writeable buffer at least the size of MinGPNVRWSize.
0Ah	*SecurityKey*	8 bytes	Varies	Security Key.

E.4 Data *Structure Element Definitions:*

type **-** Specifies the type of Data (ID) to be written.

TYPE DEFINITION TABLE				
Description	***type***	***length***	***idDATA***	***SecurityKey***
Write UUID	00h	Must be 16d bytes	FAR * to a 16d byte buffer containing the UUID	Level 3 or above. The security key is NOT required if the current UUID is blank.

Write 1394 ID	01h	Must be 8d bytes	FAR * to a 8d byte buffer containing the 1394 Unique ID.	Level 3 or above. The security key is NOT required if the current 1394ID is blank.

idData - FAR pointer to a buffer containing the actual data to be written

lpDmiWorkBuffer - FAR * to a readable / writeable buffer at least the size of *MinGPNVRWSize. MinGPNVRWSize* value can be obtained by making a Get GPNV Information (Function 55h) call.

securityKey - An 8 byte security key that meets or exceeds level 3 security (System Administrator level or above).

E.5 Return Codes:

Upon a successful call the BIOS shall return DMI_SUCCESS to the caller. If an error was made during the call, the BIOS shall return DMI_BAD_PARAMETER. If this control sub-function is not supported, the BIOS shall return DMI_INVALID_SUBFUNCTION to the caller.

If the caller does not provide a valid security key and the ID has already been written, the BIOS shall return a DMI_READ_ONLY error to the caller.

Appendix F: FAQs

Q0. Where can I find more information?

A0. More detailed information about BIOS and API services needed, and supported, by the LSA boot PROM can be found in the Network PC System Design Guidelines ver 1.0b-draft.

Q1. How does the LSA2 boot PROM begin execution?

A1. LSA2 is implemented as a standard PC/AT x86 boot PROM. It is called by the system BIOS during option ROM scan.

If it is running in a system that supports the Plug-and-Play (PnP) BIOS Boot Specification (BBS), it returns control to the system BIOS identifying itself as a valid Initial Program Load (IPL) device.

If it is not in a PnP/BBS system, it hooks interrupt vector 18h and returns control to the BIOS.

Q2. Why are there two different versions of the LSA2 for the Intel 82557/82558 based network controllers?

A2. The first version of the LSA2 (e100b.nic) is designed to be run from the boot PROM of a NIC. This code takes only 2 KB of upper memory. When the device boots, code is copied from the boot PROM into the top of free base memory.

The second version of the LSA2 (e100b.ld) is designed to be run from the BIOS boot PROM on the motherboard, like a system with built-in video. This code takes 23 KB of upper memory during option ROM scan, all but 2 KB are copied into the top of extended memory (see Q3) before control is returned to the BIOS. When the device boots, code is copied from extended memory into the top of free base memory.

Q3. Why does the BIOS lock-up during option ROM scan after running e100b.ld?

A3. Some BIOSes are not restoring the correct Global Descriptor Table (GDT) when the Protected Mode Copy (Int 15h, AH=87h) service is run during option ROM scan. These BIOSes are usually in a flat memory model during option ROM scan, and the Protected Mode Copy service returns with the processor in real-mode.

The NIC version (e100b.nic) can be used to test the rest of the LSA2 code with a BIOS that has the Protected Mode Copy bug.

To fix this bug, the BIOS needs to restore the correct GDT while in option ROM scan.

Q4. Are there any other BIOS modifications that need to be done to support LSA2?

A4. There is one (or two) Int 15h services that need to be supported if the system BIOS supports Wake-On-LAN. The first Int 15h service (AX = 2307h) returns the source of the current wake-up (this can be implemented as a destructive read, by resetting the wake-up source to be Power Switch after each read.) The second service (AH = 2308h) is used to set the source of the wake-up (this is only needed if the first service is not implemented as destructive read.)

Determine Source of Current Wake-Up

Enter:

AX := 2307h
BX := 5755h
Int 15h

Exit:

CY := 0 (Success)
AH := 00h (Success)
CL (bits 2-0) := 6 (Power Switch)

CL (bits 2-0) := 5 (LAN)
CL (bits 2-0) := 4 (COM1 RING)
CL (bits 2-0) := 3 (Timer)
CY := 1 (Failure)
AH := 86h (Unsupported function)

All other registers are unchanged.

Set Source of Next Wake-Up

Enter:

AX := 2308h
BX := 5755h
Int 15h
Exit:
CY := 0 (Success)
AH := 00h (Success)
CL (bits 7-3) := Unchanged from Determine Source of Wake-up
CL (bits 2-0) := New wake-up source (see above)
CY := 1 (Failure)
AH := 86h (Unsupported function)

All other registers are unchanged.

When setting the source of the next wake-up, you should first determine the source of the current wake-up. This ensures that bits 7-3 in CL are set correctly for the set source of wake-up service.

Q5. How is LSA2 PROM selected to be the boot device?

A5. The LSA2 is a standard PCI/PnP option ROM.

If the LSA2 is placed into a BIOS that supports PnP/BBS (BIOS Boot Specification), the BIOS should insert a network boot device into the boot order list.

If the LSA2 is placed into a BIOS that does not support PnP/BBS, that BIOS must support network devices that hook interrupt vector 18h. After the LSA2 returns control to the BIOS (at the end of the option ROM initialization call), the BIOS should check to see if Int 18h has been changed. If it has, the BIOS should assume that a network boot device has hooked Int 18h and give the user the ability to select network boot in the CMOS setup. Normally, a BIOS gives the user the ability to boot A:, C: and CD-ROM. It now needs to add network to the list of boot devices.

Appendix G: PXE PDK LSA Code Revisions

PXE PDK Release Date	PXE PDK Version	LSA Code Version	Network PC Design Guidelines Version	LSA Code Changes
9/5/97	V1.3	v.98i	V1.0b	**Bug Fixes: Functional Problems** **UNDI Driver** • Added code to shutdown NIC if media test fails. Interrupt vector table was not being restored upon exit due to media test fail. **Loader** • LOM version of loader no longer corrupts $PMM memory tables when copying itself into the allocated memory. • Corrected stack bug in loader. Systems that did not support Int 15h AX=E820h would hang. **Base Code** • Added code to use gateway IP address from DHCP ACK packet if the gateway IP address in the BINL REPLY packet is zero. • MTFTP now quits if it gets a "file not found" error. It used to try to TFTP the missing file. • 'ciaddr' field in DHCP request message is no longer filled in. This allows LSA2 to work with the Netware DHCP server. **Bug Fixes: Due to non-compliance w/ Network PC DG spec.** **Base Code** • The CLIENT_ARCHITECTURE field in DHCP DISCOVER and BINL REQUEST packets was changed from one byte, to two bytes, in network order per the Network PC DG spec.. **New functionality per Network PC DG spec. V1.0b** **UNDI Driver** • Added GET_NIC_TYPE call to UNDI so OS setup programs can determine the type of underlying NIC. --- ***Below changes are not required for PXE compliance*** **Proposed Network PC DG new functionality** **UNDI Driver** • Added GET_IFACE_INFO call to UNDI API so drivers can get information about H/W interface; specifically, so universal drivers can determine network media type (Ethernet, Token Ring, etc.) At the moment, this proposed change is only important for implementations for non-Ethernet network adapters. **Bug Fixes: Functional Problems in non-compliant legacy systems** **Loader** • Fixed hang on machine when user did not press <Space>. The BIOS required the option ROM to return control by restoring the interrupt 19h bootstrap and jumping to the 'PC/AT Compatible' BIOS location of F000:E6F2h. **Enhancements:** **Loader** • LSA2 loader now only allocates memory using $PMM, or by reducing the size of extended memory reported by Int 15h AH=88h. This is to remove a conflict between our code and Soft-ICE.

PXE PDK Release Date	PXE PDK Version	LSA Code Version	Network PC Design Guidelines Version	LSA Code Changes
				Base Code • Reduced stored DHCP packets to minimum size (548 bytes). Reduces size of run-time memory by ~3 Kbytes. • Added code to read UUID from BIOS_INFORMATION structure as documented in the "System Management BIOS Reference Specification v2.1".
7/31/97	V1.2	v.98a	V1.0b	• Added option negotiation for TSIZE. • Added new TFTP API call to get file size using TSIZE. • Number of retries in MTFTP and TFTP reduced from 8 to 4. • t_jmpbuf structure used in LSA2 library (setjmp() and longjmp() calls) did not have enough fields. The structure checksum was overwriting the CPU flags field at the end of the t_jmpbuf structure. • The s_lsa structure in the LSA2 option ROM initialization loader did not have storage reserved for the MLID initialized data field. • LSA2 did not boot in legacy systems that only supported interrupt 19h bootstrap. • TFTP/UDP packets were not being received in protected-mode on some NICs. • TFTP read API was not resetting the buffer size if the TFTP read failed. • Now recognizes PXE DHCP and ProxyDHCP packets that contain a bootfile. BINL communication is skipped. • Increases the 'seconds' field in successive DHCP packets. This enables some DHCP relay agents to forward the DHCP packets.
6/20/97	V1.1	v.97	V1.0a	• Added code to locate 'best' location for extended memory copy of LSA2 using known BIOS extended memory size services. Int 15h w/ AX=E820h, AX=E801h, AH=C7h, AH=8Ah, AX=DA88h & AH=88h. • Removed all debug output when in protected-mode. Fixes protection fault in protected-mode. • Added TFTP/UDP open/read clean-up code when an error occurs. • Moved all variables used in XMIT ISR into the base-code data segment. Fixes protection fault in protected-mode. • Commented out POST Memory Manager allocations because of internal stack bug in loader. • Corrected protected-mode segment register assignments. • Added MODE_SWITCH API call so caller can change to/from protected-mode. • Rewrite of core BOOTP/BINL packet processing to support SuperDHCP and SuperProxy servers. • Moved PXE Entry Point structure from base-code to MLID. • Added code to read and restore packet timer to fix ARP timeout bug. • Added checks for zero length segments when loader is allocating and copying base-code and MLID. • Added code to loader to handle MLID initialized data allocation and copying. • New fields were added to the TFTP_OPEN and TFTP_READ API parameter structures. • Completed multicast TFTP support. • Added IP address filters in UPD_READ so different TFTP servers could send different files with the same multicast IP address. • Corrected UNDI transmit block pointer type flag. • Added code to compute physical address of TFTP buffer for protected-mode receiving. • Removed unused error messages to save space.

PXE PDK Release Date	PXE PDK Version	LSA Code Version	Network PC Design Guidelines Version	LSA Code Changes
5/2/97	V1.0	v.92	V1.0	First distribution.

Appendix H: PXE PDK Services Revisions

PXE PDK Release Date	PXE PDK Version	Service Code Changes
9/5/97	**V1.3**	**MTFTP** • Increased the Time-To-Live field for Multicast packets to allow them to pass through routers.
		BINL • N/A
		ProxyDHCP • N/A
7/31/97	**V1.2**	**MTFTP** • Performance improvements. • Negotiation added for file size and buffer size.
		BINL • Improved packet parsing to recognize illegal packets, such as LSA2 V0.94,and ignore them • Corrected number of bytes sent for packet size to include terminating 0XFF flag
		ProxyDHCP • Initial Release.